

**Claims:**

**1. (Previously Presented)** A software architecture implemented at least in part by a computing device for a distributed computing system comprising:

a plurality of applications configured to handle requests submitted by remote devices over a network, wherein the plurality of applications are written in different programming languages;

an application program interface to present functions used by the plurality of applications to access network and computing resources of the distributed computing system; and

a common language runtime layer that translates the plurality of applications written in different programming languages into an intermediate language, the intermediate language being:

executed natively by the common language runtime layer; and

configured to access resources or services requested by the remote devices, whereby a seamless integration between multi-language application development is allowed and a robust and secure execution environment for multiple programming languages is provided.

**2. (Previously Presented)** The software architecture as recited in claim 1, wherein the distributed computing system comprises client devices and server devices that handle requests from the client devices, the remote devices comprising at least one client device.

**3. (Previously Presented)** The software architecture as recited in claim 1, wherein the distributed computing system comprises client devices and server devices that handle requests from the client devices, the remote devices comprising at least one server device that is configured as a Web server.

**4. (Previously Presented)** The software architecture as recited in claim 1, wherein the application program interface comprises:

- a first group of services related to creating Web applications;
- a second group of services related to constructing client applications;
- a third group of services related to data and handling XML documents; and
- a fourth group of services related to base class libraries.

**5. (Previously Presented)** An application program interface embodied on one or more computer readable storage media, comprising:

a first group of services related to creating Web applications;  
a second group of services related to constructing client applications;  
a third group of services related to data and handling XML documents; and  
a fourth group of services related to base class libraries; and  
a common language runtime layer that translates Web applications written in different programming languages into an intermediate language, the intermediate language being:

executed natively by the common language runtime layer; and  
configured to access resources or services, whereby a seamless integration between multi-language application development is allowed and a robust and secure execution environment for multiple programming languages is provided, wherein the seamless integration allows for the ability to use a particular code module written in a first programming language with a code module written in a second programming language.

**6. (Previously Presented)** The application program interface as recited in claim 5, wherein the first group of services comprises:

first functions that enable construction and use of Web services;  
second functions that enable temporary caching of frequently used resources;  
third functions that enable initial configuration;  
fourth functions that enable creation of controls and Web pages;  
fifth functions that enable security in Web server applications; and

sixth functions that enable access to session state values.

**7. (Previously Presented)** The application program interface as recited in claim 5, wherein the second group of services comprises:

first functions that enable creation of windowing graphical user interface environments; and

second functions that enable graphical functionality.

**8. (Previously Presented)** The application program interface as recited in claim 5, wherein the third group of services comprises:

first functions that enable management of data from multiple data sources; and

second functions that enable XML processing.

**9. (Previously Presented)** The application program interface as recited in claim 5, wherein the fourth group of services comprises:

first functions that enable definitions of various collections of objects;

second functions that enable programmatic access to configuration settings and handling of errors in configuration files;

third functions that enable application debugging and code execution tracing;

fourth functions that enable customization of data according to cultural related information;

fifth functions that enable input/output of data;

sixth functions that enable a programming interface to network protocols;  
seventh functions that enable a managed view of types, methods, and fields;  
eighth functions that enable creation, storage and management of various culture-specific resources;  
ninth functions that enable system security and permissions;  
tenth functions that enable installation and running of services;  
eleventh functions that enable character encoding;  
twelfth functions that enable multi-threaded programming; and  
thirteenth functions that facilitate runtime operations.

**10. (Original)** A network software architecture comprising the application program interface as recited in claim 5.

**11. (Previously Presented)** A distributed computer software architecture implemented at least in part by a computing device, comprising:

one or more applications written in different programming languages and configured to be executed on one or more computing devices, the one or more applications written in different programming languages handling requests submitted from remote computing devices;

a networking platform to support the one or more applications;

an application programming interface to interface the one or more applications with the networking platform; and

a common language runtime layer that translates the one or more applications written in different programming languages into an intermediate language being executed natively by the common runtime layer and configured to access resources or services requested by the remote devices, whereby a seamless integration between the one or more applications developed with multiple programming languages and the computing device is provided.

**12. (Previously Presented)** The distributed computer software architecture as recited in claim 11, further comprising a remote application configured to be executed on one of the remote computing devices, the remote application using the application programming interface to access the networking platform.

**13. (Previously Presented)** The distributed computer software architecture as recited in claim 11, wherein the application programming interface comprises:

- a first group of services related to creating Web applications;
- a second group of services related to constructing client applications;
- a third group of services related to data and handling XML documents; and
- a fourth group of services related to base class libraries.

**14. (Previously Presented)** The distributed computer software architecture as recited in claim 11, wherein the application programming interface exposes multiple functions comprising:

first functions that enable construction and use of Web services;

second functions that enable temporary caching of frequently used resources;

third functions that enable initial configuration;

fourth functions that enable creation of controls and Web pages;

fifth functions that enable security in Web server applications; and

sixth functions that enable access to session state values.

**15. (Previously Presented)** The distributed computer software architecture as recited in claim 11, wherein the application programming interface exposes multiple functions comprising:

first functions that enable creation of windowing graphical user interface environments; and

second functions that enable graphical functionality.

**16. (Previously Presented)** The distributed computer software architecture as recited in claim 11, wherein the application programming interface exposes multiple functions comprising:

first functions that enable management of data from multiple data sources; and

second functions that enable XML processing.

**17. (Previously Presented)** The distributed computer software architecture as recited in claim 11, wherein the application programming interface exposes multiple functions comprising:

- first functions that enable definitions of various collections of objects;
- second functions that enable programmatic access to configuration settings and handling of errors in configuration files;
- third functions that enable application debugging and code execution tracing;
- fourth functions that enable customization of data according to cultural related information;
- fifth functions that enable input/output of data;
- sixth functions that enable a programming interface to network protocols;
- seventh functions that enable a managed view of loaded types, methods, and fields;
- eighth functions that enable creation, storage and management of various culture-specific resources;
- ninth functions that enable system security and permissions;
- tenth functions that enable installation and running of services;
- eleventh functions that enable character encoding;
- twelfth functions that enable multi-threaded programming; and
- thirteenth functions that facilitate runtime operations.

**18. (Previously Presented)** A computer system comprising:

one or more microprocessors; and

one or more software programs that are written in different programming languages and utilize an application program interface to request services from an operating system through a common language runtime layer, the application program interface comprising:

separate commands to request services consisting of the following groups of services:

A. a first group of services related to creating Web applications, the first group of services comprising:

constructing Web services;  
temporary caching resources;  
performing initial configuration;  
creating controls and Web pages;  
enabling security in Web server applications; and  
accessing session state values;

B. a second group of services related to constructing client applications, the second group of services comprising:

creating windowing graphical user interface environments;  
and  
enabling graphical functionality;

C. a third group of services related to data and handling XML documents, the third group of services comprising:

enabling management of data from multiple data sources; and  
second functions that enable XML processing.

D. a fourth group of services related to base class libraries, the fourth group of services comprising:

defining various collections of objects;  
accessing configuration settings and handling errors in configuration files;  
debugging and tracing code execution;  
customizing data according to cultural related information;  
inputting and outputting of data;  
enabling a programming interface to network protocols;  
viewing loaded types, methods, and fields;  
creating, storing and managing various culture-specific resources;  
enabling system security and permissions;  
installing and running services;  
enabling character encoding;  
enabling multi-threaded programming; and  
facilitating runtime operations; and

a common language runtime layer that allows seamless multi-language development, with cross language inheritance and translates the one or more software programs written in different programming languages into an intermediate language, wherein the intermediate language is executed natively by the common language runtime layer and is configured to access the services requested by the one or more software programs.

**19. (Previously Presented)** A system comprising:

one or more microprocessors; and

a memory storing one or more software programs comprising computer-executable instructions executable by the one or more microprocessors, the one or more software programs comprising:

means for exposing a first set of functions that enable browser/server communication;

means for exposing a second set of functions that enable drawing and construction of client applications;

means for exposing a third set of functions that enable connectivity to data sources and XML functionality; and

means for exposing a fourth set of functions that enable system and runtime functionality; and

means for translating Web applications written in different programming languages into an intermediate language,

the intermediate language being:

executed natively by a common language runtime layer; and

configured to access resources or services, whereby a seamless integration between multi-language application development is allowed and a robust and secure execution environment for multiple programming languages is provided,

wherein the seamless integration allows for the ability to use a particular code module written in a first programming language with a code module written in a second programming language.

**20. (Previously Presented)** The system as recited in claim 19, wherein the first set of functions comprises:

first functions that enable construction and use of Web services;

second functions that enable temporary caching of frequently used resources;

third functions that enable initial configuration;

fourth functions that enable creation of controls and Web pages;

fifth functions that enable security in Web server applications; and

sixth functions that enable access to session state values.

**21. (Previously Presented)** The system as recited in claim 19, wherein the second set of functions comprises:

first functions that enable creation of windowing graphical user interface environments; and

second functions that enable graphical functionality.

**22. (Previously Presented)** The system as recited in claim 19, wherein the third set of functions comprises:

first functions that enable management of data from multiple data sources; and

second functions that enable XML processing.

**23. (Previously Presented)** The system as recited in claim 19, wherein the fourth set of functions comprises:

first functions that enable definitions of various collections of objects;

second functions that enable programmatic access to configuration settings and handling of errors in configuration files;

third functions that enable application debugging and code execution tracing;

fourth functions that enable customization of data according to cultural related information;

fifth functions that enable input/output of data;

sixth functions that enable a programming interface to network protocols;

seventh functions that enable a managed view of loaded types, methods, and fields;

eighth functions that enable creation, storage and management of various culture-specific resources;

ninth functions that enable system security and permissions;

tenth functions that enable installation and running of services;

eleventh functions that enable character encoding;

twelfth functions that enable multi-threaded programming; and

thirteenth functions that facilitate runtime operations.

**24. (Previously Presented)** A method implemented at least in part by a computer, comprising:

managing network and computing resources for a distributed computing system;

exposing a set of functions that enable developers to access the network and computing resources of the distributed computing system, the set of functions comprising first functions to facilitate browser/server communication, second functions to facilitate construction of client applications, third functions to facilitate connectivity to data sources and XML functionality, and fourth functions to access system and runtime resources; and

providing a common language runtime layer that allows seamless multi-language development, with cross language inheritance and translates Web applications written in different programming languages into an intermediate language that is supported by the

common language runtime layer and is configured to access the network and computing resources requested by the developers, whereby a seamless integration provides a robust and secure execution environment for multiple programming languages.

**25. (Previously Presented)** The method as recited in claim 24, further comprising receiving a request from a remote computing device, the request containing a call to at least one of the first, second, third, and fourth functions.

**26. (Previously Presented)** A method implemented at least in part by a computer, the method comprising:

creating a first namespace with functions that enable browser/server communication;

creating a second namespace with functions that enable drawing and construction of client applications;

creating a third namespace with functions that enable connectivity to data sources and XML functionality;

creating a fourth namespace with functions that enable system and runtime functionality; and

providing a common language runtime layer that translates Web applications written in different programming languages into an intermediate language, the intermediate language being:

executed natively by the common language runtime layer; and

configured to access resources or services requested by the client applications, whereby a seamless integration between multi-language application development is allowed and a robust and secure execution environment for multiple programming languages is provided.

**27. (Previously Presented)** The method as recited in claim 26, wherein the first namespace defines classes that facilitate:

- construction and use of Web services;
- temporary caching of resources;
- initial configuration;
- creation of controls and Web pages;
- security in Web server applications; and
- access to session state values.

**28. (Previously Presented)** The method as recited in claim 26, wherein the second namespace defines classes that facilitate:

- creation of windowing graphical user interface environments; and
- graphical functionality.

**29. (Previously Presented)** The method as recited in claim 26, wherein the third namespace defines classes that facilitate:

management of data from multiple data sources; and  
processing of XML documents.

**30. (Previously Presented)** The method as recited in claim 26, wherein the fourth namespace defines classes that facilitate:

programmatic access to configuration settings and handling of errors in configuration files;

application debugging and code execution tracing;

customization of data according to cultural related information;

inputting and outputting of data;

interfacing to network protocols;

viewing loaded types, methods, and fields;

creation, storage and management of various culture-specific resources;

system security and permissions;

installation and running of services;

character encoding;

multi-threaded programming; and

runtime operations.

**31. (Previously Presented)** A method implemented at least in part by a computer, the method comprising:

calling one or more first functions to facilitate browser/server communication;

calling one or more second functions to facilitate construction of client applications;

calling one or more third functions to facilitate connectivity to data sources and XML functionality;

calling one or more fourth functions to access system and runtime resources; and

using a common language runtime layer that translates Web applications written in different programming languages into an intermediate language that is:

executed natively by the common language runtime layer; and

configured to access resources or services requested by the client applications, whereby a seamless integration between multi-language application development is allowed and a robust and secure execution environment for multiple programming languages is provided, wherein the seamless integration allows for the ability to use a particular code module written in a first programming language with a code module written in a second programming language.

**32. (Previously Presented)** The method as recited in claim 31, wherein the first functions comprise functions for construction and use of Web services, temporary caching of resources, initial configuration, creation of controls and pages that will appear as user interfaces, securing Web server applications, and accessing session state values.

**33. (Previously Presented)** The method as recited in claim 31, wherein the second functions comprise functions for creation of windowing graphical user interface environments, and graphical functionality.

**34. (Previously Presented)** The method as recited in claim 31, wherein the third functions comprise functions for management of data from multiple data sources, and XML processing.

**35. (Previously Presented)** The method as recited in claim 31, wherein the fourth functions comprise functions for programmatic access to configuration settings, application debugging and code execution tracing, customization of text according to cultural related information, synchronous and asynchronous reading from and writing to data streams and files, creation and management of various culture-specific resources, system security and permissions, installation and running of services, character encoding, and multi-threaded programming.

**36. (Previously Presented)** A method implemented at least in part by a computer, the method comprising:

receiving one or more calls to one or more first functions to facilitate browser/server communication;

receiving one or more calls to one or more second functions to facilitate construction of client applications;

receiving one or more calls to one or more third functions to facilitate connectivity to data sources and XML functionality;

receiving one or more calls to one or more fourth functions to access system and runtime resources; and

using a common language runtime layer that allows seamless multi-language development, with cross language inheritance and translates Web applications written in different programming languages into an intermediate language that is supported by the common language runtime layer and configured to access services requested by the client applications, whereby a seamless integration provides a robust and secure execution environment for multiple programming languages .

**37. (Previously Presented)** The method as recited in claim 36, wherein the first functions comprise functions for construction and use of Web services, temporary caching of resources, initial configuration, creation of controls and pages that will appear as user interfaces, securing Web server applications, and accessing session state values.

**38. (Previously Presented)** The method as recited in claim 36, wherein the second functions comprise functions for creation of windowing graphical user interface environments, and graphical functionality.

**39. (Previously Presented)** The method as recited in claim 36, wherein the third functions comprise functions for management of data from multiple data sources, and XML processing.

**40. (Previously Presented)** The method as recited in claim 36, wherein the fourth functions comprise functions for programmatic access to configuration settings, application debugging and code execution tracing, customization of text according to cultural related information, synchronous and asynchronous reading from and writing to data streams and files, creation and management of various culture-specific resources, system security and permissions, installation and running of services, character encoding, and multi-threaded programming.

**41. (Currently Amended)** A method implemented at least in part by a computer, for exposing resources using an application program interface, the method comprising:

exposing a first group of services related to creating Web applications, the first group of services comprising:

- constructing Web services;
- temporary caching resources;
- performing initial configuration;
- creating controls and Web pages;
- enabling security in Web server applications; and
- accessing session state values;

exposing a second group of services related to constructing client applications, the second group of services comprising:

- creating windowing graphical user interface environments; and
- enabling graphical functionality;

exposing a third group of services related to data and handling XML documents, the third group comprising:

- enabling management of data from multiple data sources; and
- second functions that enable XML processing.

exposing a fourth group of services related to base class libraries, the fourth group of services comprising:

- defining various collections of objects;

accessing configuration settings and handling errors in configuration files;  
debugging and tracing code execution;  
customizing data according to cultural related information;  
inputting and outputting of data;  
enabling a programming interface to network protocols;  
viewing loaded types, methods, and fields;  
creating, storing and managing various culture-specific resources;  
enabling system security and permissions;  
installing and running services;  
enabling character encoding;  
enabling multi-threaded programming; and  
facilitating runtime operations; and

providing a common language runtime layer that translates Web applications written in different programming languages into an intermediate language,

the intermediate language being:

executed natively by the common language runtime layer; and

configured to access resources requested by the client applications;

wherein, the different ~~program~~ programming languages are selected from a plurality of programming languages, the plurality of programming languages comprising:

Visual Basic;

C++;

C#;

COBOL;

Jscript;

Perl;

Eiffel; and

Python.